Modul Praktikum 4. Issue dalam K-Means Clustering

1. Penentuan jumlah Cluster dengan Elbow Method

Salah satu isu yang harus ditentukan ketika menggunakan K means Clustering adalah menentukan jumlah Cluster untuk data yang ada. Praktikum ini mempelajari bagaimana menentukan Jumlah Cluster optimal dengan menggunakan Elbow Method.

a) Elbow Method

Definisi:

"The elbow method is used to determine the optimal number of clusters in k-means clustering. The elbow method plots the value of the cost function produced by different values of k."

b) Experiment

i. Importing library

```
%matplotlib inline
#%matplotlib inline sets the backend of matplotlib to the 'inline' backend
import matplotlib.pyplot as plt
#Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy
plt.style.use('seaborn-whitegrid')
#Seaborn style on matplotlib plot, menentukan jenis graph. untuk jenis graph lain bisa dilihat di
#https://python-graph-gallery.com/199-matplotlib-style-sheets/
import numpy as np
from sklearn.cluster import KMeans
```

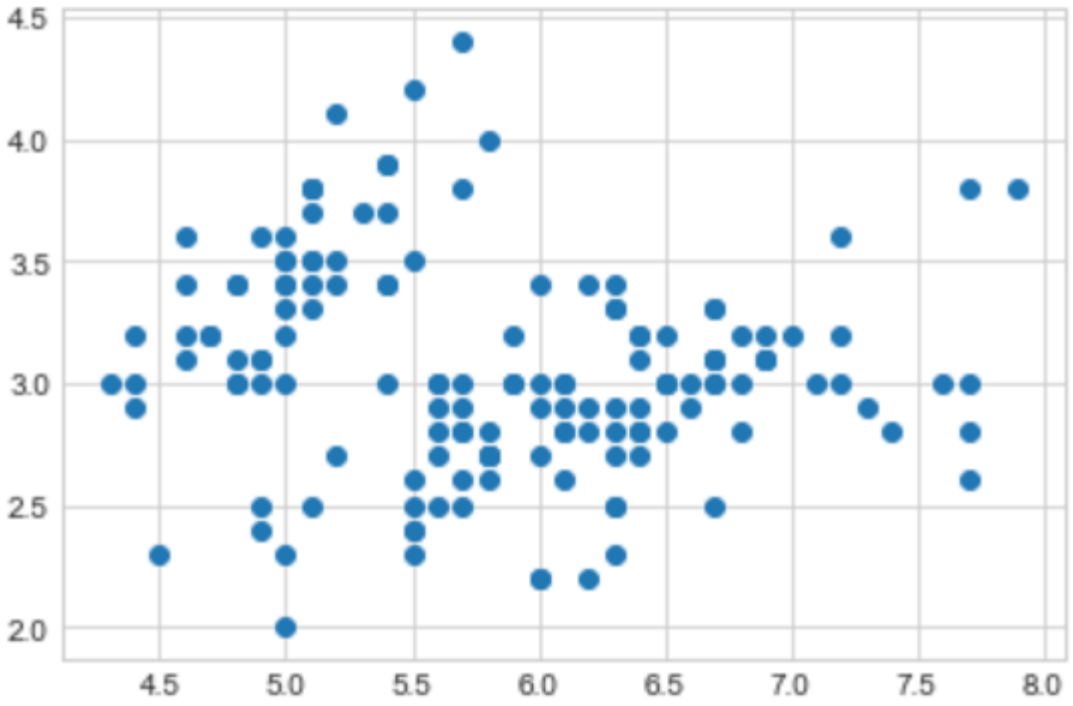
Cobalah dengan menggunakan jenis graph lain yang tersedia di matplotlib style sheets.

ii. Data

Data yang digunakan dalam eksperimen ini adalah data iris. Dataset ini terdiri atas empat fitur yaitu Sepal Length, Sepal Width, Petal Length, dan Petal Width. Serta terdiri atas tiga kelas, yaitu Iris-setosa, Iris-versicolor, dan Iris-virginica. Pada Iris dataset, jumlah keseluruhan data adalah 150 baris dengan masing-masing kelas

memiliki 50 baris data. Dalam eksperimen ini kita akan memakai 2 fitur dari data iris, tanpa menggunakan label dari data tersebut untuk melakukan clustering. Data iris terdiri dari data pengukuran fitur (iris.data) dan label (iris.target_names)

```
# membuat data set
from sklearn.datasets import load_iris
iris = load_iris()
features=iris.data.T
#mengambil data dari fitur data iris.
plt.plot()
plt.scatter(features[0], features[1]) #2 fitur yang akan dipakai
plt.show()
#catatan: jika ingin mencetak/mmengetahui data iris, di baris paling
bawah bisa diketik iris. (atau iris.data , atau iris.target_names)
```



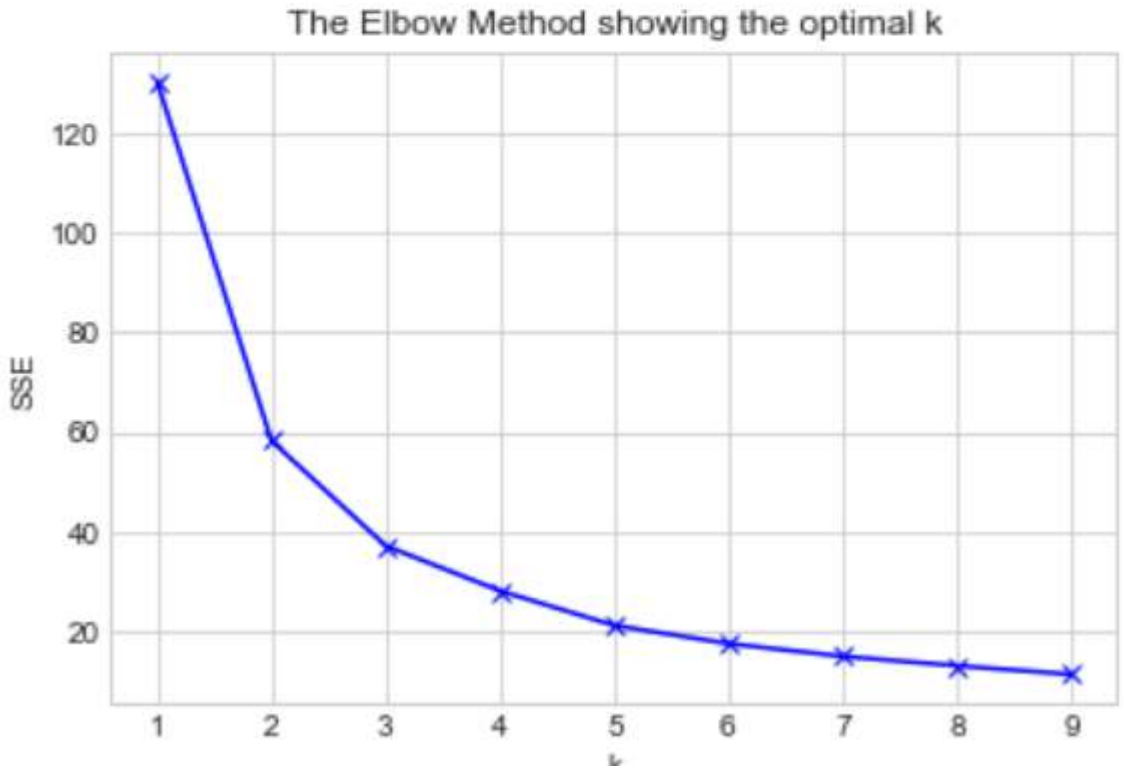
iii. Menentukan nilai yang tepat untuk cluster

Untuk menentukan nilai yang tepat untuk K, kita dapat menghitung Sum of Squared error dari cluster yang dihasilkan. Dalam python hal tersebut dapat dilakukan dengan menyimpan nilai inertia dalam variable SSE sebagai berikut

```
X = np.array(list(zip(features[0], features[1]))).reshape(len(features[1]), 2)
SSE = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k).fit(X)
    kmeanModel.fit(X)
    SSE.append(kmeanModel.inertia_)
```

iv. Plot nilai K

```
# Plot the elbow
plt.plot(K, SSE, 'bx-')
plt.xlabel('k')
plt.ylabel('SSE')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```

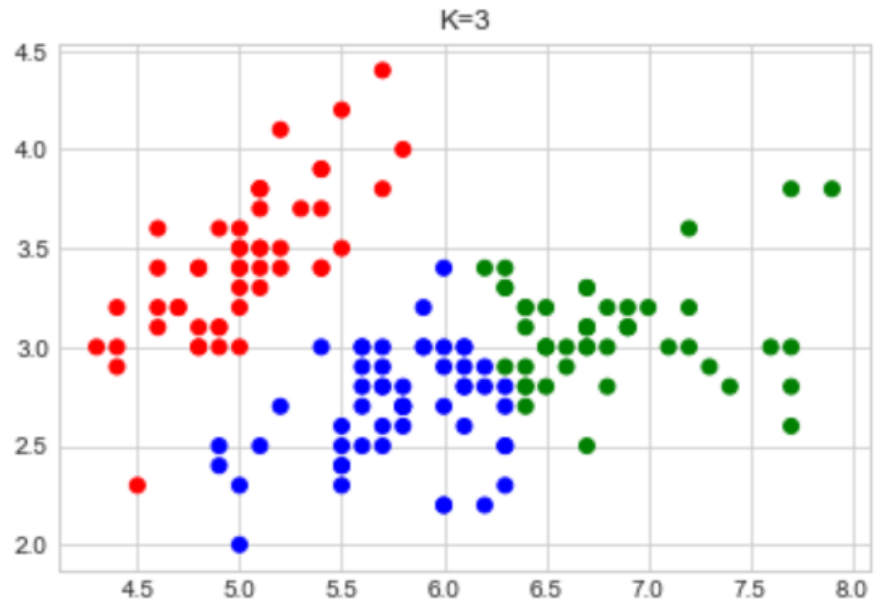


Catatan: gantilah 'bx-' dengan x- dan x. Apa yang anda amati? Berdasar pengamatan anda apa fungsi dari variable tersebut?

v. Menentukan nilai K berdasar elbow graph tersebut.

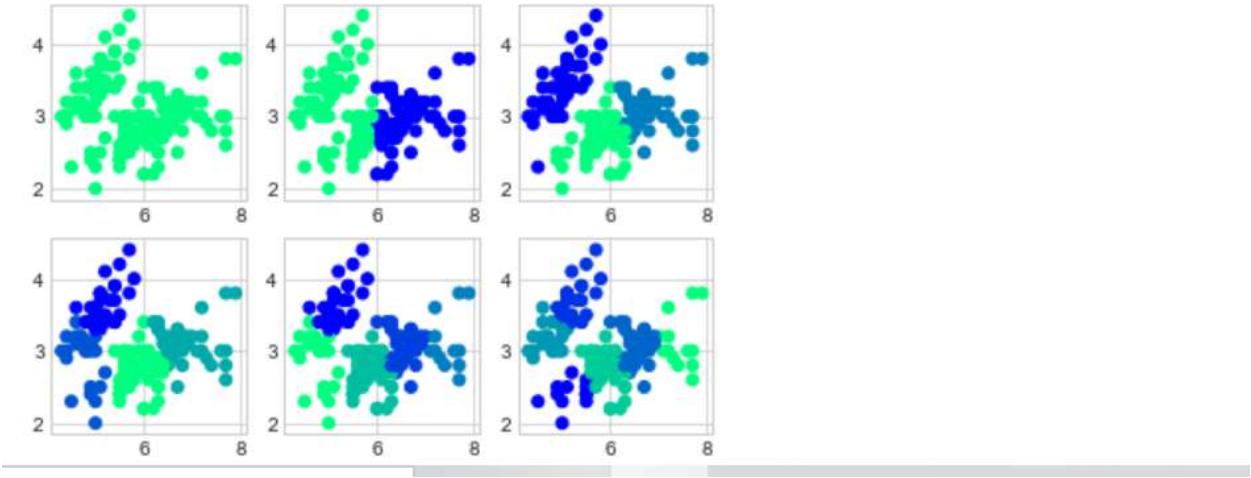
Berdasar nilai K yang anda dapatkan, anda dapat melakukan clustering dengan nilai K tersebut. Sebagai contoh jika kita ambil K adalah 3 maka:

```
y_pred = KMeans(n_clusters=3).fit_predict(X)
plt.plot
LABEL_COLOR_MAP = {0 : 'r',
                   1 : 'g',
                   2 :'b'
                   }

label_color = [LABEL_COLOR_MAP[l] for l in y_pred]
plt.scatter(features[0],features[1], c=label_color)
plt.title("K=3")
plt.show()
```



PErcobaan tambahan dapat dilakukan untuk melihat bentuk cluster untuk semua kemungkinan K yang kita inginkan. Misalkan ketika kita ingin melihat hasil cluster untuk K dari 1 sampai 6 maka :

```
figure,ax=plt.subplots(2,3)
K=range(1,7)
for k in K:
    if(k<4):
        row=0
        column=k-1
    else:
        row=1
        column=k-4
    kmeanModel = KMeans(n_clusters=k).fit(X)
    y_pred = kmeanModel.fit_predict(X)
    ax[row][column].scatter(features[0],features[1], c=y_pred,cmap='winter_r')

plt.show()
```



Catatan: plt.subplots: Create a figure and a set of subplots. Mengembalikan 2 parameter figure dan axis (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.subplots.html)

Plt.subplots(2,3) berarti membuat subplot 2 baris dan 3 kolom.

Dalam mengisi gambar di subplot membutuhkan informasi pada baris dan keberapa yang ingin diisi. Oleh karena itu ax[row][column] menyatakan hal tersebut. Nilai row dan column ditentukan oleh nilai k. jika nilai k 1 maka berada di baris 1 kolom 1.

Plt.scatter plot of *y* vs *x* with varying marker size and/or color. Dalam hal ini x dan y adalah feature[0] dan feature[1]. Untuk masing-masing plot maka label dibedakan dengan warna. Pewarnaan ini sesuai dari y_pred yang didapatkan dengan melakukan prediksi dengan hasil clustering sesuai dengan K yang dipakai. (c=y_pred)
https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.scatter.html

Sementara itu cmap adalah color map digunakan untuk membedakan pewarnaan pada scatter. Beberapa contoh color map yang lain bisa dilihat di https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html

Penilaian:

1. Gunakan fitur 2 dan 3 dalam data iris untuk menentukan jumlah K dengan elbow method
2. Plot hasil cluster anda dengan nilai K yang anda pilih!
3. Tunjukkan dengan data tersebut, hasil clustering untuk K 1 sampai 6 dengan colormap "coolwarm". Tunjukkan juga dengan 3 color map lain yang tersedia di web tersebut!!

2. Inisialisasi titik awal k means clustering

Salah satu permasalahan dalam K means clustering adalah kemungkinan mendapatkan hasil yang tidak sesuai yang kita inginkan jika melakukan random initialization. Dalam hal ini, python sudah menghandle dengan menetapkan variable init bernilai default Kmeans++.

K-Means clustering

Read more in the User Guide.

Parameters:

n_clusters : *int, optional, default: 8*

The number of clusters to form as well as the number of centroids to generate.

init : *{'k-means++', 'random' or an ndarray}*

Method for initialization, defaults to 'k-means++':

'k-means++' : selects initial cluster centers for k-mean clustering in a smart way to speed up convergence. See section Notes in k_init for more details.

'random': choose k observations (rows) at random from data for the initial centroids.

If an ndarray is passed, it should be of shape (n_clusters, n_features) and gives the initial centers.

n_init : *int, default: 10*

Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.

max_iter : *int, default: 300*